University of Verona

Machine learning & artificial intelligence

Professor: Vittorio Murino

Hierarchical Text Classification

Author:
Theodora Poultsidou

June 24, 2024

Contents

1 Motivation And Rationale

In this project, I am performing text classification, which is a common task in Natural Language Processing (NLP). Although it may appear simple, it is quite challenging to implement, compare, and fine-tune different classifiers.

To make the project more interesting, I chose a dataset with a hierarchical structure. This will allow me to compare the performance of models with and without utilizing the hierarchy. Since hierarchical text classification is not very common, it presents an interesting challenge to explore.

Additionally, I will be implementing both machine learning and deep learning classifiers. While deep learning often outperforms machine learning in image processing, I am curious to see how it performs with text. By comparing these two approaches, I hope to determine if deep learning models can indeed outperform machine learning models in this hierarchical text classification task.

2 Objectives

2.1 General Objective

The primary objective of this project is to develop an effective hierarchical text classification system that can classify short text into multiple levels of categories and subcategories. To achieve this, I will implement various machine learning and deep learning classifiers. I'll start with basic models like Naive Bayes and Logistic Regression, move on to more advanced methods like Support Vector Machines (SVC) and K-Nearest Neighbors (KNN), and finally, explore deep learning models, specifically Convolutional Neural Networks (CNNs).

2.2 Specific Objectives

2.2.1 Implement Baseline Models

1. **Naive Bayes**: Starting with Naive Bayes due to its simplicity and effectiveness in text classification tasks, I will explore different values of alpha to understand its impact on performance.

2. **Logistic Regression**: Evaluating Logistic Regression to see its ability to handle multi-class classification and analyzing the effect when varying the regularization parameter C.

2.2.2 Implement Advanced Machine Learning Models

1. **Support Vector Machines (SVC)**: Exploring Support Vector Machines (SVC), well-known for their strength in classification tasks, and experimenting again with different values of the regularization parameter C.

2. **K-Nearest Neighbors (KNN)**: Using this model to understand the impact of different values of K on classification performance.

2.2.3 Explore Deep Learning Models

1. **Convolutional Neural Networks (CNNs)**: Given their well-known success in image classification, I will explore CNNs to try and improve the classification accuracy when working with text and compare them to the machine learning classifiers to see if they outperform them. To achieve the best results, I will experiment with different parameters and kernels because guessing the correct ones on the first try would be too easy! However, I will not report these details extensively, but will instead explain the final model.

2.2.4 Taking Advantage of Hierarchical Structures

1. Leveraging the hierarchical relationship between the categories and the subcategories, aiming to improve the classification performace.

2. Compare the performance of models with and without hierarchical information to assess the benefits.

2.2.5 Evaluate and Compare

1. Evaluate each model using standard metrics such as accuracy, precision, recall, and F1-score.
2. Create plots to compare the performance of different hyperparameters for each model.
3. Specifically analyze how the inclusion of hierarchical structures impacts the classification performance of subcategories and the model's complexity.

2.3 Considerations and Questions

1. Is there a possibility that deep learning models, like Convolutional Neural Networks, might not always outperform common machine learning classifiers in hierarchical text classification tasks?
2. Does the hierarchical structure add complexity and computational cost to text classification? And do its benefits outweigh the potential for error propagation?

3 Methodology

3.1 Dataset Selection

The dataset that is used in this project is a collection of news articles, each labeled with multiple categories and subcategories.

More specific: [from the description of the dataset]
The dataset consists of 10,917 news articles with hierarchical news categories collected between 1 January 2019 and 31 December 2019. The articles are manually labeled based on a hierarchical taxonomy with 17 first-level and 109 second-level categories. This dataset can be used to train machine learning models for automatically classifying news articles by topic.

It includes the following columns:

- **data_id**: Unique identifier of the data entry.
- **id**: Unique identifier of the article.
- **date**: Date of the article release.
- **source**: Publisher information of the article.
- **title**: Title of the news article.
- **content**: Text of the news article.
- **author**: Author of the news article.
- **url**: Link to the original article.
- **published**: Date of article publication in local time.
- **published_utc**: Date of article publication in UTC time.
- **collection_utc**: Date of article scraping in UTC time.
- **category_level_1**: First level category of Media Topic NewsCodes's taxonomy.
- **category_level_2**: Second level category of Media Topic NewsCodes's taxonomy.

3.2 Data Preprocessing

3.2.1 Loading the Data

The dataset will be loaded using pandas to start with the analysis.

3.2.2 Text Cleaning and Tokenization

The text cleaning process is essential for preparing data for effective machine/deep learning tasks, confirming the principle ”garbage in, garbage out.”

Firstly, I merge the ”title” and ”content” features into a single ”text” feature, which I use as the single predictor, dropping all other columns as I found them not necessary for my analysis. Once I have the ”text” feature, I proceed with some essential cleaning steps.

- **Removing Irrelevant Characters**: Eliminate any non-alphanumeric characters such as punctuation marks, special symbols, and numbers that do not add value to the text's meaning. This helps in reducing noise from the data and focuses on the meaningful and more important words.
- **Lowercasing**: Convert the text to lowercase ensures that words like ”The” and ”the” are treated the same, reducing the redundancy and improving the consistency of the text data.
- **Stop Words Removal**: Stop words are common words like ”and,” ”the,” ”is,” etc., that do not carry significant meaning and can be removed to improve the efficiency of the text processing. Using the NLTK library for this purpose, which provides a comprehensive list of stop words. Additionally, I manually select and remove certain words specific to our dataset that do not contribute to the classification task.
- **Tokenization**: This step involves breaking down the cleaned text into individual words or tokens. Tokenization is necessary because machine learning models typically work with numerical representations of these tokens rather than raw text.
- **Stemming**: Involves reducing words to their base forms by removing case and ilection information. This approach not only simplifies words but also enhances the effectiveness of document representation, which is constructed by concatenating stem counts.

3.2.3 Vectorization

WE use a method called TF-IDF (Term Frequency-Inverse Document Frequency) to turn the text into numerical vectors. This technique helps to point out the important words in a document while decreasing the importance of words that appear frequently in many documents. The idea is that if a word shows up in lots of documents, it's probably not very unique or useful for distinguishing between different categories. To make sure the length of a document doesn't affect our analysis, we also normalize each vector to have the same length. This makes the data consistent for the machine learning algorithms to process.

3.2.4 Splitting the Dataset

To evaluate the models, I splitted the dataset into training and testing sets using an 80-20 ratio. Additionally,I used 5-fold cross-validation to ensure consistent evaluation and to see if the model generalizes.

3.3 Algorithms and Models

Now, we will explore several machine learning and deep learning models that we will evaluate their performance in hierarchical text classification. Since this is a multi-class classification problem, I'll apply the One vs Rest (OvR) strategy to which model can not manage the multi-class natively(Logistic Regression, SVC with 'linear' kernel). OvR works by training one classifier per class, treating the samples of that class as positive and all other samples as negative. This helps the model handle multi-class classification effectively , transforming it into multiple binary ones.

3.3.1 Baseline Models

Naive Bayes :

- Naive Bayes is a probabilistic classifier based on Bayes' theorem, which is suitable for text classification because it's simple and efficient.
- I'll use the MultinomialNB because it handles the multiclass classification without the need of the OvR and vary the alpha parameter to prevent overfitting and underfitting, while keeping other parameters at their default settings.

Logistic Regression:

- Logistic Regression is a linear model for binary classification that can be extended to multi-class problems using the OvR classifier.
- I'll use the LogisticRegression classifier and vary the regularization parameter C to study its impact on performance.

3.3.2 Advanced Machine Learning Models

Support Vector Machines (SVC):

- SVC is a robust classifier when it comes to text classification that aims to find the optimal hyperplane separating different classes.
- I'll use the SVC classifier with the 'linear' kernel , wrapped by the OvR strategy and vary the regularization parameter C to find the best performance.

K-Nearest Neighbors (KNN) :

- KNN is a non-parametric method used for classification by finding the most common labels among the k-nearest neighbors.
- I'll use the KNeighborsClassifier and explore different values of k to understand its impact on classification performance.

3.3.3 Deep Learning Models

Convolutional Neural Networks (CNNs):

- The CNN we used has a very common build:
  * **Embedding Layer**: Turns words into dense vectors that capture their meanings.
  * **Convolutional Layers**: Extract features from the text using filters and ReLU activation.
  * **Max-Pooling Layers**: Reduce the size of the feature maps while keeping the important information.
  * **Dropout Layers**: We used a dropout rate of 0.2 to prevent overfitting.
  * **Fully Connected (Dense) Layers**: Combine features learned by the convolutional layers.
  * **Output Layer**: Uses a softmax activation to produce the probabilities for each class.
- I used cross-entropy as the loss function, which is standard for multi-class classification.
- I used Early Stopping to prevent overfitting by stopping training when the validation loss stops improving.

3.4 Leveraging the Hierarchy

One of the objectives is to leverage the hierarchical relationship between categories and subcategories and aim to improve classification performance. We achieve this by first classify each instance into its respective Level 1 category, and then subsequently classify it within Level 2 subcategories specific to that Level 1 category.

This approach is helpful because it breaks down the classification task into more manageable steps, allowing the model to focus on classifying between fewer, more closely related categories at each stage. This hierarchical method can enhance the accuracy and efficiency, mainly in complex datasets with a large number of categories.

3.5 Model Training and Evaluation

Each model will be trained on the training set, with some of the hyperparameters tuned using cross-validation to prevent overfitting. I will use k-fold cross-validation with k = 5 for both category level 1 and category level 2 to ensure best evaluation of the models. But when I will establish the hierarchical stucture, I will not cross validate, because it was observed data leakage, meaning that data meant for the test set was present in the training set and in reverse. To avoid this I will train and evaluate on a single split.

Models will be evaluated using standard metrics such as accuracy, precision, recall, and F1-score. Additionally, plots will be created to visualize the performance of each model with different hyperparameters.

3.6 Analytical and Computational Tools

3.6.1 Programming Language

Python: The primary language for implementing the models and preprocessing the data due to its extensive libraries for machine learning and data manipulation.

3.6.2 Libraries and Frameworks

- **Pandas**: For data loading and manipulation.
- **Scikit-learn**: For implementing machine learning models such as Naive Bayes, Logistic Regression, SVC, and KNN.
- **TensorFlow/Keras**: For building and training the CNN.
- **NLTK/Spacy**: For the text preprocessing tasks such as tokenization and stop-word removal.
- **Matplotlib/Seaborn**: For creating visualizations to compare model performances.

4 Experiments & Results

4.1 Naive Bayes Classifier

After testing the Naive Bayes model with different alpha parameters, we found the optimal alpha value to be 0.1, as is confirmed by the plot below. The plot shows the training and testing accuracy for different values of alpha. The model performed best with an alpha value of 0.1, achieving a cross-validated accuracy of 0.7416 for Level 1 categories.

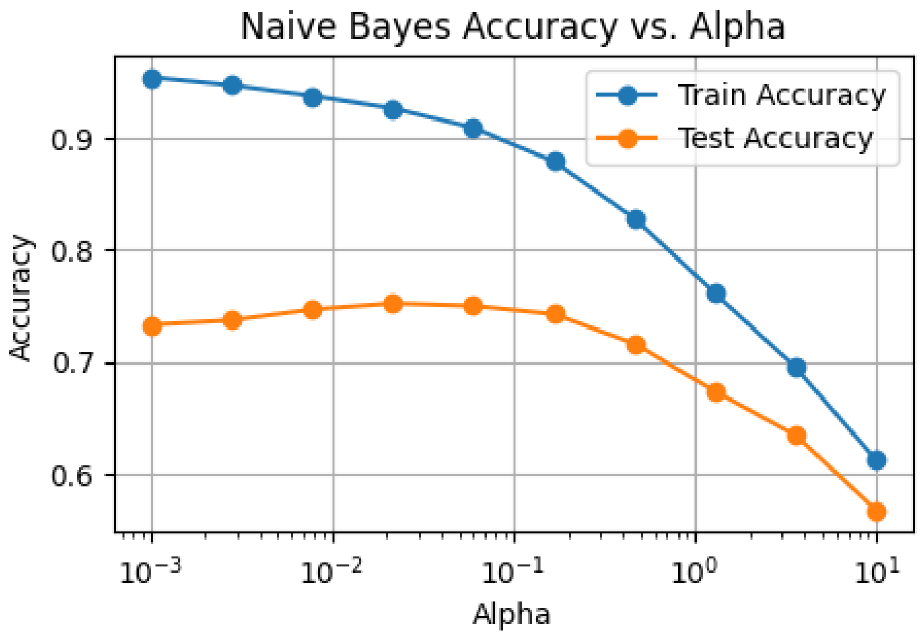


Figure 1: Naive Bayes Accuracy vs. Alpha

Observation:

1. From the plot, we can observe the variance-bias trade-off. As the alpha value increases, the training accuracy decreases significantly, that indicates that the model is becoming more generalized and less overfitted to the training data (reduction in variance). However, the test accuracy also starts to drop after a certain point, showing that too much generalization leads to underfitting (increase in bias). The optimal alpha value of 0.1 provides a balance where both training and test accuracies are relatively high, suggesting a good trade-off between bias and variance.

Note: The testing on different values of alpha was done only on Level 1 categories. For Level 2 categories, which contain a lot more categories, we used the same alpha value of 0.1.

Here are all the results for Level 1 categories, Level 2 subcategories, and the hierarchical Level 2 subcategories.

Metric	Level 1 Categories	Level 2 Categories	Hierarchy Level 2
Accuracy	0.7416	0.5840	0.7370
Precision	0.7713	0.5938	0.7392
Recall	0.7276	0.5839	0.7368
F1-Score	0.7372	0.5759	0.7252

Table 1: Metrics for Optimal Alpha (= 0.1)

Observations:

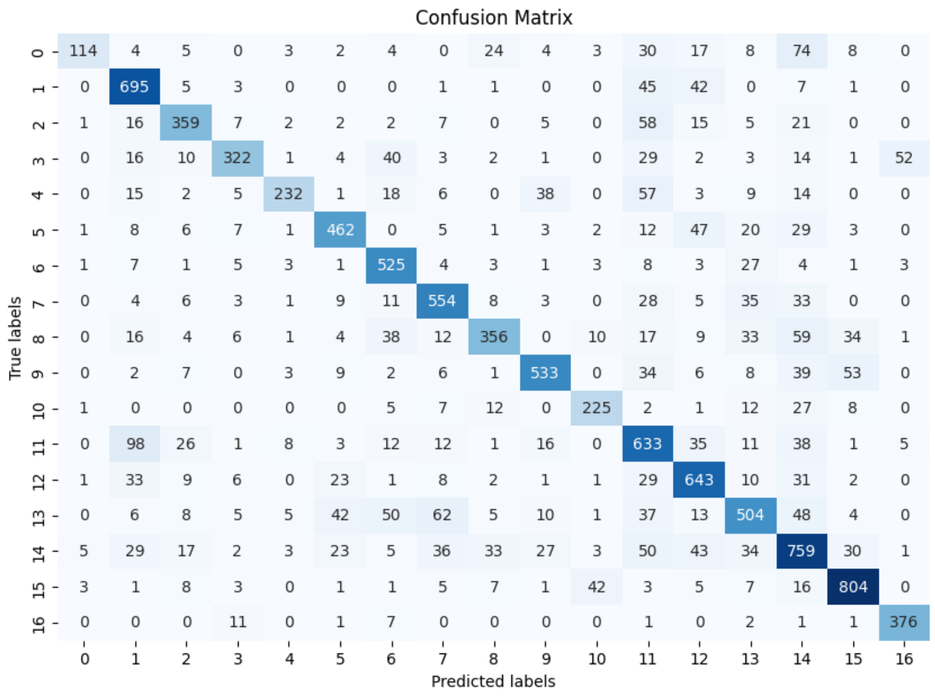
1. The metrics for Level 1 categories are descently high, indicating a good performance and suggesting that the Naive Bayes Multinomial classifier is effective for Level 1 categories (17 in total).

2. For Level 1, we observe a balance between precision and recall, indicating that the model is able to identify similar categories with a relatively good trade-off between false positives and false negatives.

3. Moving to the Level 2 categories, the metrics are much lower compared to Level 1, which was expected as the categories are ten times more numerous, increasing the complexity. (A slightly higher alpha, such as 1, might be more suitable?)

4. We can clearly see a significant increase(around 15%) in the metrics for Level 2 categories after incorporating the hierarchical structure, highlighting the benefits of leveraging the hierarchical information.

I am presenting the confusion matrix for Level 1 only, as the Level 2 confusion matrix would be 109x109, which is too large and unclear to interpret.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	114	4	5	0	3	2	4	0	24	4	3	30	17	8	74	8	0
1	0	695	5	3	0	0	0	1	1	0	0	45	42	0	7	1	0
2	1	16	359	7	2	2	2	7	0	5	0	58	15	5	21	0	0
3	0	16	10	322	1	4	40	3	2	1	0	29	2	3	14	1	52
4	0	15	2	5	232	1	18	6	0	38	0	57	3	9	14	0	0
5	1	8	6	7	1	462	0	5	1	3	2	12	47	20	29	3	0
6	1	7	1	5	3	1	525	4	3	1	3	8	3	27	4	1	3
7	0	4	6	3	1	9	11	554	8	3	0	28	5	35	33	0	0
8	0	16	4	6	1	4	38	12	356	0	10	17	9	33	59	34	1
9	0	2	7	0	3	9	2	6	1	533	0	34	6	8	39	53	0
10	1	0	0	0	0	0	5	7	12	0	225	2	1	12	27	8	0
11	0	98	26	1	8	3	12	12	1	16	0	633	35	11	38	1	5
12	1	33	9	6	0	23	1	8	2	1	1	29	643	10	31	2	0
13	0	6	8	5	5	42	50	62	5	10	1	37	13	504	48	4	0
14	5	29	17	2	3	23	5	36	33	27	3	50	43	34	759	30	1
15	3	1	8	3	0	1	1	5	7	1	42	3	5	7	16	804	0
16	0	0	0	11	0	1	7	0	0	0	0	1	0	2	1	1	376

Figure 2: Confussion Matrix for Level 1

Observations:

1. As we wanted, we observed high numbers along the diagonal, indicating that the model was correctly predicting the majority of instances for each category.

2. Although, there are some off the diagonal instances that are being misclassified.

4.2 Logistic Regression Classifier

Continuing with the Logistic Regression, we tuned the model using different values for the C parameter (regularization) to observe the impact of regularization on the model's performance. The plot below shows the training and testing accuracy for different values of C. We tested C values ranging from 0.001 to 100, and the optimal value was found to be 100, achieving a cross-validated accuracy of 0.8099 for Level 1 categories.

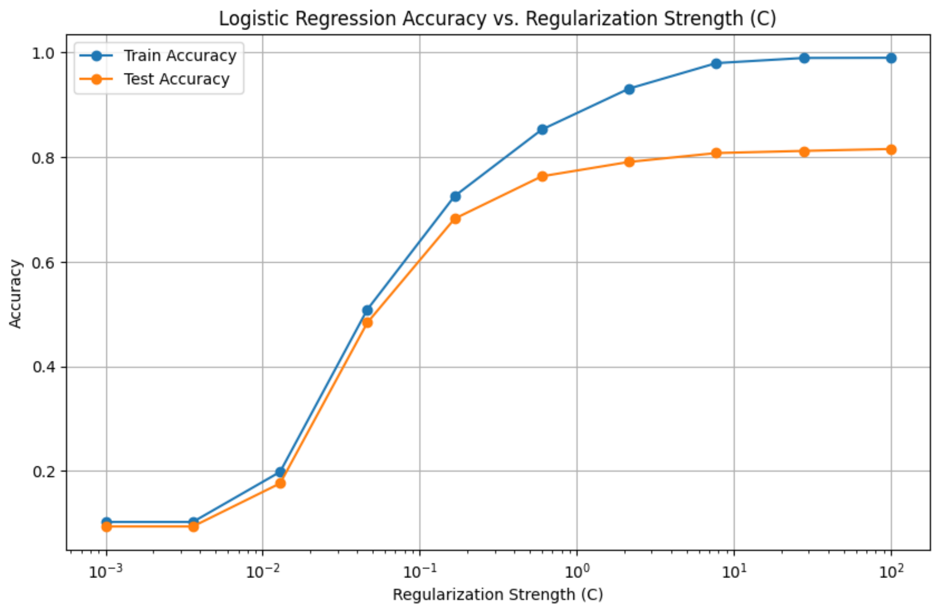


Figure 3: Logistic Regression Accuracy vs. Regularization Strength C

Observation:

1. From the plot, we can observe the impact of regularization on the model's performance. As the value of C increases, the regularization decreasses and the model fits the training data more closely. Thus, both the training and test accuracies improve, indicating low bias.

The results for Level 1 categories, Level 2 subcategories, and the hierarchical Level 2 subcategories.

Metric	Level 1 Categories	Level 2 Categories	Hierarchy Level 2
Accuracy	0.8099	0.6656	0.7807
Precision	0.8173	0.6600	0.7751
Recall	0.8072	0.6656	0.7788
F1-Score	0.8116	0.6603	0.7701

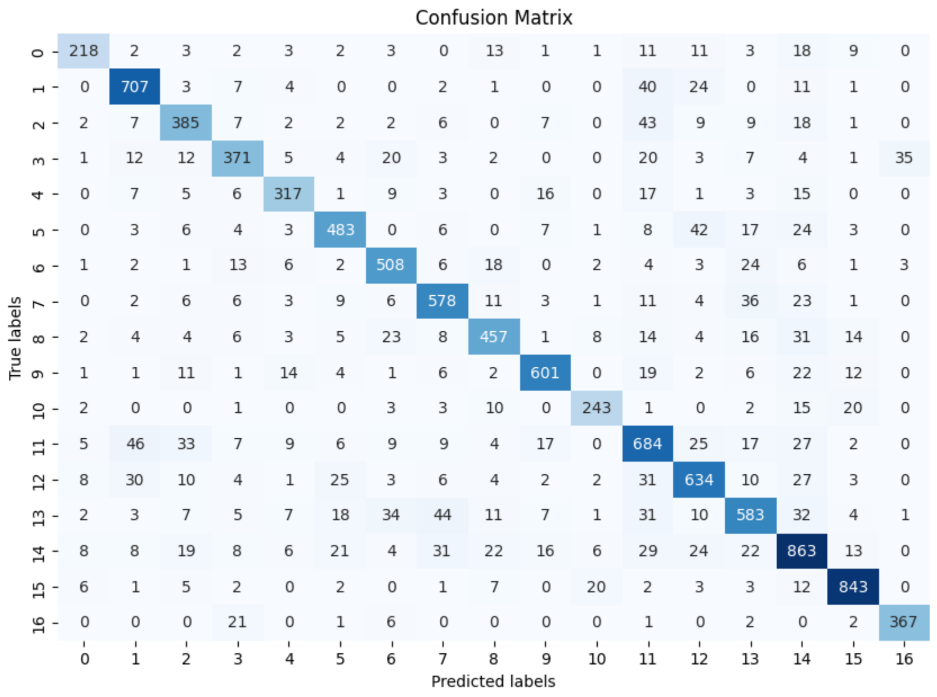
Table 2: Metrics for C = 100

Observations:

1. The Logistic Regression model achieves pretty high accuracy, precision, recall, and F1-score for Level 1 categories, indicating strong performance and balance.

2. Incorporating the hierarchical structure all the metrics for Level 2 categories significantly improved, compared to using Level 2 categories alone.

Confussion Matrix:


Confusion Matrix
True labels
Predicted labels

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	218	2	3	2	3	2	3	0	13	1	1	11	11	3	18	9	0
1	0	707	3	7	4	0	0	2	1	0	0	40	24	0	11	1	0
2	2	7	385	7	2	2	2	6	0	7	0	43	9	9	18	1	0
3	1	12	12	371	5	4	20	3	2	0	0	20	3	7	4	1	35
4	0	7	5	6	317	1	9	3	0	16	0	17	1	3	15	0	0
5	0	3	6	4	3	483	0	6	0	7	1	8	42	17	24	3	0
6	1	2	1	13	6	2	508	6	18	0	2	4	3	24	6	1	3
7	0	2	6	6	3	9	6	578	11	3	1	11	4	36	23	1	0
8	2	4	4	6	3	5	23	8	457	1	8	14	4	16	31	14	0
9	1	1	11	1	14	4	1	6	2	601	0	19	2	6	22	12	0
10	2	0	0	1	0	0	3	3	10	0	243	1	0	2	15	20	0
11	5	46	33	7	9	6	9	9	4	17	0	684	25	17	27	2	0
12	8	30	10	4	1	25	3	6	4	2	2	31	634	10	27	3	0
13	2	3	7	5	7	18	34	44	11	7	1	31	10	583	32	4	1
14	8	8	19	8	6	21	4	31	22	16	6	29	24	22	863	13	0
15	6	1	5	2	0	2	0	1	7	0	20	2	3	3	12	843	0
16	0	0	0	21	0	1	6	0	0	0	0	1	0	2	0	2	367

Figure 4: Logistic Regression Accuracy vs. Regularization Strength C

4.3 Support Vector Classifier (SVC)

As before with the Logistic Regression, we tuned the SVC classifier with different regularization parameters and observed the optimal one to be $C = 10$. We tested C values using the range C_values $= 2^{-5}$ to 2^{15}, as is commonly suggeseted in the literature to test the regularization parameter C. In the plot below, the x-axis is in log scale.

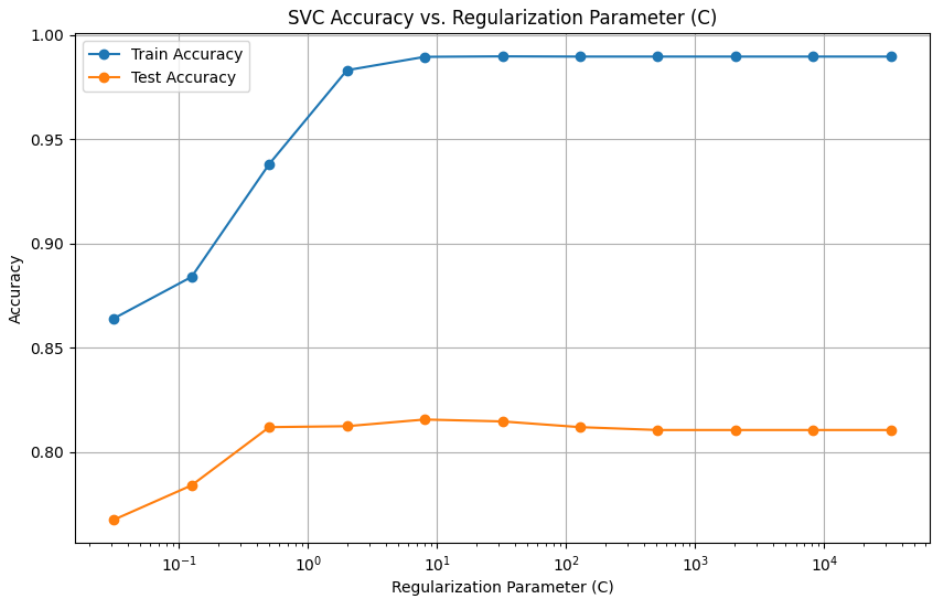


Figure 5: SVC Accuracy vs. Regularization Parameter (C)

Observation:

1. From the plot, we can observe that as the value of C increases, the training accuracy continues to improve, indicating reduced bias. However, the test accuracy peaks around $C = 10$ (which corresponds to 10^1 in log scale) and then starts to plateau, suggesting this as the optimal value for regularization.

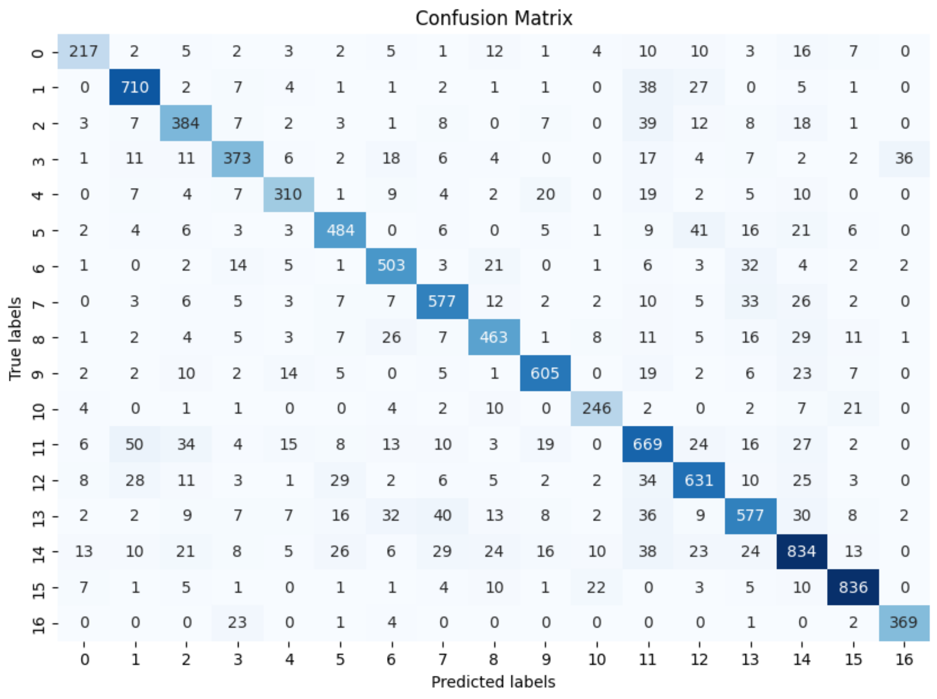
Here are all the metrics for every classification for optimal C=10

Metric	Level 1 Categories	Level 2 Categories	Hierarchy Level 2
Cross-Validated Accuracy	0.8050	0.6666	0.7739
Cross-Validated Precision	0.8089	0.6616	0.7742
Cross-Validated Recall	0.8040	0.6667	0.7705
Cross-Validated F1-Score	0.8060	0.6619	0.7659

Table 3: Metrics for Optimal C

.

The Confusion Matrix for Level 1:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	217	2	5	2	3	2	5	1	12	1	4	10	10	3	16	7	0
1	0	710	2	7	4	1	1	2	1	1	0	38	27	0	5	1	0
2	3	7	384	7	2	3	1	8	0	7	0	39	12	8	18	1	0
3	1	11	11	373	6	2	18	6	4	0	0	17	4	7	2	2	36
4	0	7	4	7	310	1	9	4	2	20	0	19	2	5	10	0	0
5	2	4	6	3	3	484	0	6	0	5	1	9	41	16	21	6	0
6	1	0	2	14	5	1	503	3	21	0	1	6	3	32	4	2	2
7	0	3	6	5	3	7	7	577	12	2	2	10	5	33	26	2	0
8	1	2	4	5	3	7	26	7	463	1	8	11	5	16	29	11	1
9	2	2	10	2	14	5	0	5	1	605	0	19	2	6	23	7	0
10	4	0	1	1	0	0	4	2	10	0	246	2	0	2	7	21	0
11	6	50	34	4	15	8	13	10	3	19	0	669	24	16	27	2	0
12	8	28	11	3	1	29	2	6	5	2	2	34	631	10	25	3	0
13	2	2	9	7	7	16	32	40	13	8	2	36	9	577	30	8	2
14	13	10	21	8	5	26	6	29	24	16	10	38	23	24	834	13	0
15	7	1	5	1	0	1	1	4	10	1	22	0	3	5	10	836	0
16	0	0	0	23	0	1	4	0	0	0	0	0	0	1	0	2	369

Figure 6: Confusion Matrix for SVC

4.4 K-Nearest Neighbors (KNN)

For the K-Nearest Neighbors (KNN) classifier, I tuned the model using different values for the number of neighbors (k) to observe its impact on the model's performance. The plot below shows the training and testing accuracy for different values of k.

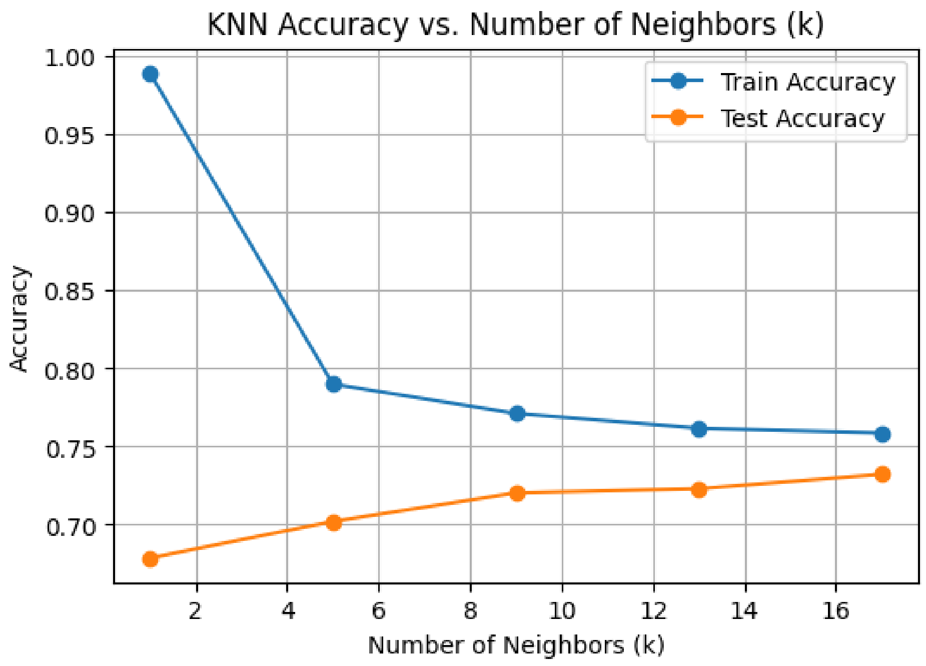


Figure 7: KNN Accuracy vs. Number of Neighbors (k)

Observation:

1. From the plot, we can observe that as the number of neighbors (k) increases, the training accuracy decreases, indicating that the model is becoming less overfitted. Conversely, the test accuracy increases and then stabilizes, suggesting an optimal k value that balances the bias and the variance.

Note: The optimal k value was found to be around 15. The testing on different values of k was done only on Level 1 categories.

Here are all the results for Level 1 categories, Level 2 subcategories, and the hierarchical Level 2 subcategories.

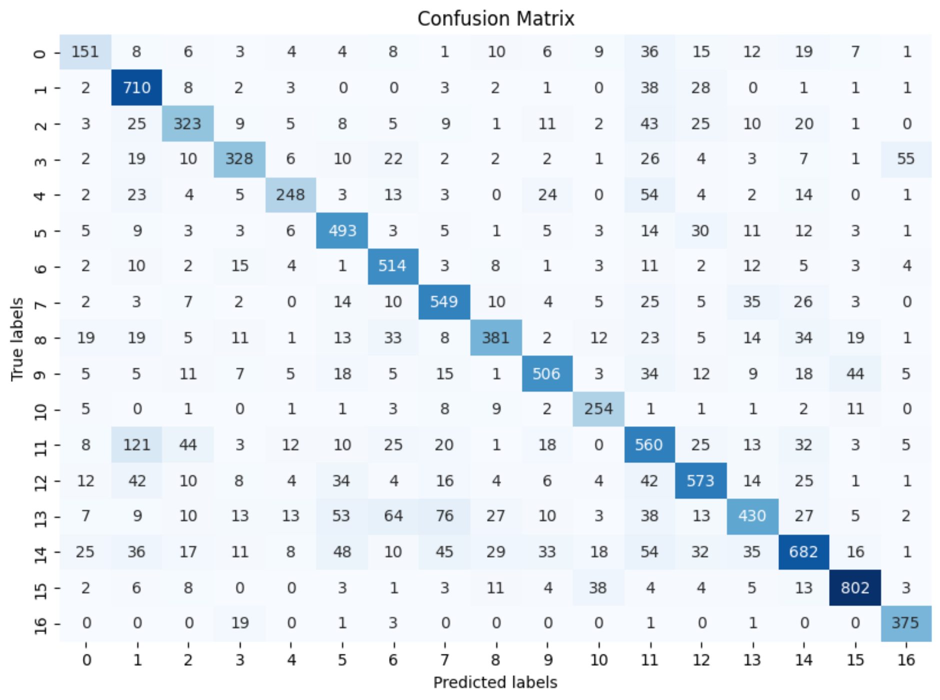
Metric	Level 1 Categories	Level 2 Categories	Hierarchy Level 2
Accuracy	0.7217	0.5278	0.7061
Precision	0.7242	0.5222	0.7077
Recall	0.7227	0.5304	0.7036
F1-Score	0.7188	0.5121	0.6940

Table 4: Metrics for Optimal k

Observations:

1. Observations similar with the previous models.

The Confussion Matrix for KNN on Level 1 categories:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	151	8	6	3	4	4	8	1	10	6	9	36	15	12	19	7	1
1	2	710	8	2	3	0	0	3	2	1	0	38	28	0	1	1	1
2	3	25	323	9	5	8	5	9	1	11	2	43	25	10	20	1	0
3	2	19	10	328	6	10	22	2	2	2	1	26	4	3	7	1	55
4	2	23	4	5	248	3	13	3	0	24	0	54	4	2	14	0	1
5	5	9	3	3	6	493	3	5	1	5	3	14	30	11	12	3	1
6	2	10	2	15	4	1	514	3	8	1	3	11	2	12	5	3	4
7	2	3	7	2	0	14	10	549	10	4	5	25	5	35	26	3	0
8	19	19	5	11	1	13	33	8	381	2	12	23	5	14	34	19	1
9	5	5	11	7	5	18	5	15	1	506	3	34	12	9	18	44	5
10	5	0	1	0	1	1	3	8	9	2	254	1	1	1	2	11	0
11	8	121	44	3	12	10	25	20	1	18	0	560	25	13	32	3	5
12	12	42	10	8	4	34	4	16	4	6	4	42	573	14	25	1	1
13	7	9	10	13	13	53	64	76	27	10	3	38	13	430	27	5	2
14	25	36	17	11	8	48	10	45	29	33	18	54	32	35	682	16	1
15	2	6	8	0	0	3	1	3	11	4	38	4	4	5	13	802	3
16	0	0	0	19	0	1	3	0	0	0	0	1	0	1	0	0	375

Figure 8: Confusion Matrix KNN for Level 1

Observations:

1. Same observations as before.

4.5 Convolutional Neural Networks (CNNs)

For the Convolutional Neural Networks (CNNs) classifier, we trained the model over several epochs, establishing Early Stopping to prevent overfitting. We observed its performance in terms of training and test accuracy and loss. The plots below show the training and validation accuracy and loss over the epochs for the Level 1 and Level 2 categories separately.

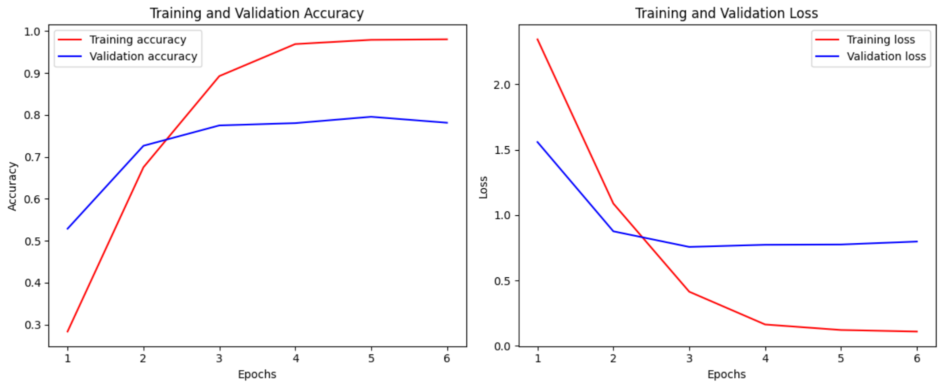


Figure 9: CNN Training and Validation Accuracy and Loss for Level 1

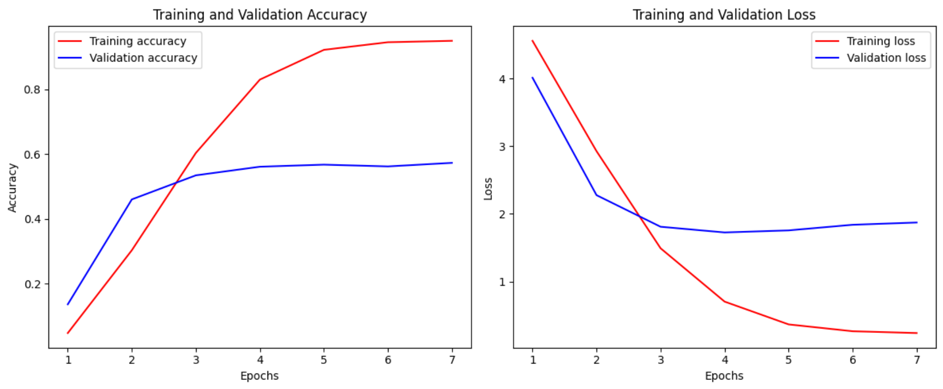


Figure 10: CNN Training and Validation Accuracy and Loss for Level 2

Observations:

1. From the plots, we can observe that the training and validation accuracies both improve over the epochs, although the validation accuracy starts to stabilize after a few epochs, indicating the model is reaching its generalization limit.

2. Similarly, the training loss decreases consistently while the validation loss decreases initially but then stabilizes, suggesting that further training might lead to overfitting, which is prevented by Early Stopping.

Here are all the results for Level 1 categories, Level 2 subcategories, and the hierarchical Level 2 subcategories.

Metric	Level 1 Categories	Level 2 Categories	Hierarchy Level 2
Accuracy	0.7147	0.4185	0.7537
Precision	0.8544	0.6760	0.7550
Recall	0.6852	0.4134	0.7425
F1-Score	0.7438	0.4855	0.7355

Table 5: Metrics for CNNs

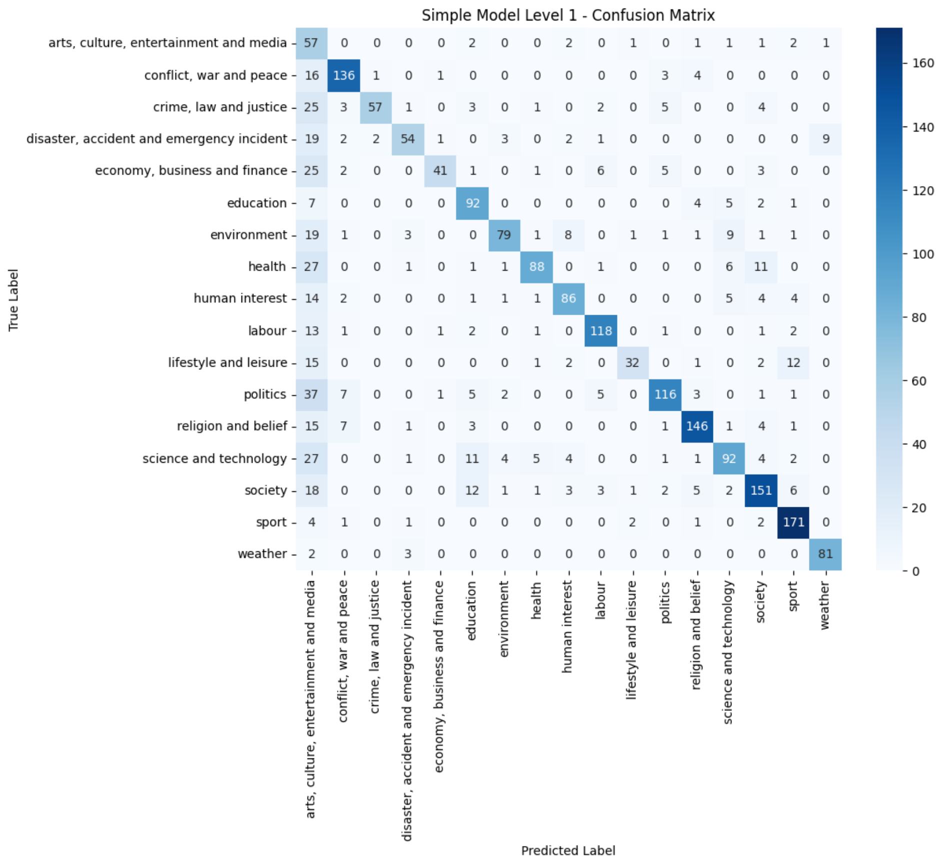
Observations:

1. The precision scores are notable higher compared to the other metrics across all levels (Level 1, Level 2, and Hierarchical Level 2). This indicates that the CNN is very confident about the instances it labels as positive.

2. The recall scores, while lower than the precision scores, are still reasonably high, indicating the model is also effective at identifying most of the similar instances, although it misses some.

3. The F1-scores, which balance precision and recall, provide a good overall measure of the model's performance. The relatively high F1-scores at Level 1 and Hierarchical Level 2 suggest that the CNN classifier performs well when hierarchical structure is utilized.

I am presenting the confusion matrix for Level 1 only, as the Level 2 confusion matrix would be too large and unclear to interpret.



True Label \ Predicted Label	arts, culture, entertainment and media	conflict, war and peace	crime, law and justice	disaster, accident and emergency incident	economy, business and finance	education	environment	health	human interest	labour	lifestyle and leisure	politics	religion and belief	science and technology	society	sport	weather
arts, culture, entertainment and media	57	0	0	0	0	2	0	0	2	0	1	0	1	1	1	2	1
conflict, war and peace	16	136	1	0	1	0	0	0	0	0	0	3	4	0	0	0	0
crime, law and justice	25	3	57	1	0	3	0	1	0	2	0	5	0	0	4	0	0
disaster, accident and emergency incident	19	2	2	54	1	0	3	0	2	1	0	0	0	0	0	0	9
economy, business and finance	25	2	0	0	41	1	0	1	0	6	0	5	0	0	3	0	0
education	7	0	0	0	0	92	0	0	0	0	0	0	4	5	2	1	0
environment	19	1	0	3	0	0	79	1	8	0	1	1	1	9	1	1	0
health	27	0	0	1	0	1	1	88	0	1	0	0	0	6	11	0	0
human interest	14	2	0	0	0	1	1	1	86	0	0	0	0	5	4	4	0
labour	13	1	0	0	1	2	0	1	0	118	0	1	0	0	1	2	0
lifestyle and leisure	15	0	0	0	0	0	0	1	2	0	32	0	1	0	2	12	0
politics	37	7	0	0	1	5	2	0	0	5	0	116	3	0	1	1	0
religion and belief	15	7	0	1	0	3	0	0	0	0	0	1	146	1	4	1	0
science and technology	27	0	0	1	0	11	4	5	4	0	0	1	1	92	4	2	0
society	18	0	0	0	0	12	1	1	3	3	1	2	5	2	151	6	0
sport	4	1	0	1	0	0	0	0	0	0	2	0	1	0	2	171	0
weather	2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	81

Figure 11: Confusion Matrix for Level 1

Observations:

1. As expected, we observe high numbers along the diagonal, indicating that the model is correctly predicting the majority of instances for each category.
2. There are some off the diagonal instances that are being misclassified, most of them corresponding to the first category, which contains the most articles. This might suggest that the CNN is biased towards predicting this category over the others .
3. Also, the numbers on the diagonal of the CNN confusion matrix are much lower compared to the other models. This could be because the CNN results are not cross-validated, while the other models's results are. In cross-validation, predictions from all the folds(5 in total) are combined into one prediction set, which is then used to make the confusion matrix.

4.6 Comparison between all the models

4.6.1 Comparison of the evaluation metrics

Metric	Naive Bayes	Logistic Regression	SVC	KNN	CNN
Level 1 Categories					
Accuracy	0.7416	0.8099	0.8050	0.7217	0.7147
Precision	0.7713	0.8173	0.8089	0.7242	0.8544
Recall	0.7276	0.8072	0.8040	0.7227	0.6852
F1-Score	0.7372	0.8116	0.8060	0.7188	0.7438
Level 2 Categories					
Accuracy	0.5840	0.6656	0.6666	0.5278	0.4185
Precision	0.5938	0.6600	0.6616	0.5222	0.6760
Recall	0.5839	0.6656	0.6667	0.5304	0.4134
F1-Score	0.5759	0.6603	0.6619	0.5121	0.4855
Hierarchy Level 2					
Accuracy	0.7370	0.7807	0.7739	0.7061	0.7537
Precision	0.7392	0.7751	0.7742	0.7077	0.7550
Recall	0.7368	0.7788	0.7705	0.7036	0.7425
F1-Score	0.7252	0.7701	0.7659	0.6940	0.7355

Table 6: Comparison of Metrics for Different Models

Observations:

1. **Level 1 Categories**:

   - Logistic Regression achieves the highest accuracy (0.8099) and F1-Score (0.8116), making it the best performer model for Level 1 categories.
   - CNN achieves the highest precision (0.8544), meaning it's really good at avoiding the false positives.
   - KNN has the lowest accuracy (0.7217) and F1-Score (0.7188), so it's not as effective for Level 1 categories compared to the other models.

2. **Level 2 Categories**:

   - SVC and Logistic Regression have similar accuracy (0.6666 and 0.6656) and F1-Score (0.6619 and 0.6603), showing they perform about the same.
   - CNN has the highest precision (0.6760), but its accuracy (0.4185) and F1-Score (0.4855) are much lower, so it struggles more with Level 2 categories.
   - KNN has the lowest scores for Level 2 categories, showing it's the least effective for these finer-grained classifications.

3. **Hierarchy Level 2**:

   - Logistic Regression again has the highest accuracy (0.7807) and F1-Score (0.7701), confirming it's great with the hierarchical structure.
   - CNN and SVC perform similarly, with CNN having a slightly higher accuracy (0.7537) but a slightly lower F1-Score (0.7355) compared to SVC.
   - KNN, while better with hierarchical information, still performs the worst among the models with an F1-Score of 0.6940.

4. **General Observations**:

   - Using hierarchical structure presents that significantly improves the metrics for Level 2 categories across all models, showing the importance and the power of hierarchical structures in text classification.

- The Naive Bayes model, even though it's not the top performer, is consistent across all levels, making it a reliable choice.
- The CNN model, despite its high precision, has lower recall, meaning it's more conservative in its predictions and might miss some relevant instances.
- The top performers are SVC and Logistic Regression. SVC, which has had a reputation s as one of the best models for text classification, once again proves its effectiveness in this task.

4.6.2 Time Complexity Comparison

To better understand the complexity of each model, we compared their training and prediction times f Level 2 categories, and Level 2 categories with hierarchy. We specifically looked at how the hierarchy benefits or increases the complexity of the models. The times for each model are as follows:

Model	Train Time	Pred Time	Train Time Hierarchy	Pred Time Hierarchy
Naive Bayes	0.3454	0.0367	0.0056	0.0014
Logistic Regression	356.7899	0.0672	1.5897	0.0006
SVC	534.5873	129.2596	1.2860	0.2600
KNN	0.0076	78.8717	0.0005	0.5518

Table 7: Training and Prediction Times for Different Levels of Categories

Observations:

1. **Naive Bayes**:
   - Naive Bayes has the shortest training and prediction times across all levels, making it extremely efficient. Specifically, the times for Level 2 with hierarchy are very low (0.0056s training, 0.0014s prediction).
2. **Logistic Regression**:
   - Logistic Regression has significantly longer training times for Level 2 categories (356.7899s), but these times are much shorter when using hierarchical information (1.5897s training). Prediction times remain quick across all levels.
3. **SVC**:
   - SVC shows the longest training times for Level 2 categories (534.5873s) and high prediction times (129.2596s). However, using hierarchical information reduces these times significantly (1.2860s training, 0.2600s prediction), though prediction time remains substantial.
4. **KNN**:
   - KNN has the shortest training times but the longest prediction times across all levels. For Level 2 with hierarchy, the training time is 0.0005s, but the prediction time is still high at 0.5518s.
5. **General Observations**:
   - SVC, while being a top performer in terms of accuracy and F1-Score, is the most computationally expensive in terms of training and prediction times, especially without hierarchical information.
   - Naive Bayes and Logistic Regression are consistently efficient in terms of both training and prediction times. Even though they already have small times, they still see significant reductions when using hierarchical information.
   - KNN and SVC show a tremendous decrease in training and prediction times when hierarchical structure is used.

5 Conclusion

In this project, we implemented and compared various machine learning and deep learning models for hierarchical text classification. The models included Naive Bayes, Logistic Regression, Support Vector Classifier (SVC), K-Nearest Neighbors (KNN), and Convolutional Neural Networks (CNNs). I evaluated their performance across Level 1 categories, Level 2 categories, and Level 2 categories with hierarchical information.

Summary of Comparisons:

1. **Top Performers**:

   - Logistic Regression and SVC were the top performers in terms of accuracy and F1-Score across different levels. SVC, known for its reputation in text classification, once again proved its effectiveness.

2. **Time Complexity**:

   - Naive Bayes and KNN were the most efficient in terms of training time, but KNN had significantly higher prediction times.
   - SVC had the highest computational cost in both training and prediction times, although it performed very well in terms of classification accuracy.
   - Logistic Regression provided a good balance between computational efficiency and performance.

3. **Hierarchical Information**:

   - Leveraging hierarchical structure significantly improved the performance metrics for Level 2 categories across all models.
   - Hierarchical structures reduced training and prediction times for models like Logistic Regression and SVC, highlighting their potential benefits in computational efficiency.

Answers to the Research Questions:

1. **Possibility of Deep Learning Models Not Always Outperforming Traditional Machine Learning Classifiers**:
   Our results showed that CNNs did not always outperform traditional machine learning classifiers like Logistic Regression and SVC in hierarchical text classification tasks. While CNNs had high precision, they struggled with recall and overall F1-Score, indicating that traditional models can still be highly competitive and sometimes superior in certain tasks.

2. **Complexity and Computational Cost of Hierarchical Structures**:
   Incorporating the hierarchical structures into text classification does not add significant complexity or computational cost, as demonstrated by our time complexity analysis. Instead, it offers substantial benefits in terms of improved classification performance and reduced prediction times for complex models like SVC and Logistic Regression. The use of hierarchical information enhances the model's ability to handle classifications effectively.

Conclusion:
In conclusion, while deep learning models like CNNs have shown great promise in various domains, traditional machine learning classifiers like Logistic Regression and SVC remain highly effective for hierarchical text classification tasks. The incorporation of hierarchical information proves to be beneficial, enhancing both performance and efficiency, and is a crucial consideration for future text classification projects.